VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2014

Jiří Draška

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Draška**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: ABB s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Pavla Dráždilová, Ph.D.**

Konzultant bakalářské práce: Ing. Radek Novák

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA OSTRAVA * Fakulta elektrotechniky a informatiky * OSTRAVA 1

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 14. 7. 2014

.....

Rád bych tímto poděkoval společnosti ABB s.r.o. za příležitost vykonat bakalářskou praxi v právě jejich společnosti. Rovněž bych chtěl poděkovat svému konzultantovi Ing. Radku Novákovi za ochotný přístup, toleranci a za rady v průběhu mé bakalářské praxe.

Abstrakt

Tato bakalářská práce pojednává o mé pracovní činnosti ve společnosti ABB s.r.o., kde probíhalo absolvování mé odborné praxe. V prvních dvou týdnech praxe jsem sbíral zkušenosti v oblasti Windows instalátorů, které měly sloužit pro ABB aplikace, jenž jsou nasazované v ropném průmyslu. Následně mi bylo změněno téma na vývoj testovacího nástroje, který slouží pro interní využití. Tento nástroj má za úkol zjednodušit práci s pluginy, jež slouží pro různá testování.

Klíčová slova: C#, .NET, WiX Toolset, WinForms, ABB s.r.o.

Abstract

This bachelor thesis desribes my work activity in ABB s.r.o. where my practice took place. In the first two weeks of practice, I gained experience in Windows installer, which should be used for ABB applications that are deployed in the oil industry. Thereafter was the topic changed on the development of test instruments that is used for internal use. This test instrument is intended to simplify the work with plugins which are used for testing.

Keywords: C#, .NET, WiX Toolset, WinForms, ABB s.r.o.

Seznam použitých zkratek a symbolů

MSDN	–	Microsoft Development Network
C#	–	Microsoft Team Foundation Server
XML	–	Extensible Markup Language
LINQ	–	Language Integrated Query
CLR	–	Common Language Runtime
.NET	–	Software framework developed by Microsoft

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

Seznam tabulek

1 Úvod

Bakalářskou praxi jsem si vybral z toho důvodu, že jsem si chtěl ověřit své znalosti ze studia a rovněž si tyto znalosti dále rozšířit i po praktické stránce. Tím, že mé zkušenosti v oblasti IT vyplývaly pouze z teoretických poznatků na vysoké škole, se mi jevila tato volba pro bakalářskou praxi jako zajímavá příležitost.

Společnost ABB s.r.o. jsem si vybral ze seznamu firem umístěného na stránkách informačního systému KatIS. Tuto firmu jsem oslovil prostřednictvím e-mailu, jehož součástí byl také můj životopis. Poté mne manažer softwarového týmu pozval na pohovor a následně mi bylo sděleno, že jsem byl přijat na pozici .NET programátor.

Na počátku praxe mi byl přiřazen projekt, který zahrnoval programování instalátorů pro kontrolní systém, jehož hlavním úkolem je monitorování alarm systémů v ropném průmyslu. Tento kontrolní systém například pomáhá monitorovat stav systému alarmu, zvýšit jeho výkon a automaticky generovat závěrečné zprávy. V tomto projektu jsem nahlédl do problematiky programování instalátorů. Naučil jsem se, jak vypadá taková struktura instalátoru, jak se konfigurují uživatelská okna a nakonec jsem si samotný instalátor naprogramoval. Po dvou týdnech praxe jsem začal intenzivněji pracovat na dalším novém tématu.

Mým druhým projektem na praxi byl projekt Testing Tool. Na tomto projektu jsem pracoval od úplného začátku a je programován v jazyce C#. Primárním cílem projektu Testing Tool je zjednodušit práci s nástroji (pluginy), jež jsou do prostředí Testing Tool nahrány a poté v něm spuštěny. Každý takový plugin je pro něco využíván, například pro testování, přičemž v prostředí Testing Tool může být spuštěno paralelně takových pluginů více. Tímto je pro uživatele zbavena nutnost "přepínat" mez jednotlivými pluginy a jejich výsledky. Testing Tool navíc poskytuje plnohodnotné uživatelské rozhraní daného pluginu.

V první kapitole této bakalářské práce se věnuji popisu odborného zaměření společnosti a mému pracovnímu zařazení. V druhé kapitole popisuji seznámení se s nástrojem Windows Installer XML Toolset (WIX), zadané úkoly, jejich řešení a časovou náročnost těchto úkolů. Třetí kapitola pojednává o práci na projektu Testing Toolu. V páté kapitole se věnuji svým scházejícím a získaným dovednostem během praxe. V závěrečné kapitole shrnuji dosažené výsledky v průběhu odborné praxe, její přínos a svůj celkový dojem.

2 Odborné zařazení firmy a popis pracovního zařazení

V této kapitole je popsáno odborné zaměření firmy, ve které jsem vykonával praxi a také mé pracovní zařazení.

2.1 Odborné zaměření firmy

ABB (Asea Brown Boveri) je přední světová společnost poskytující technologie pro energetiku a automatizaci.V České republice působí ABB svými produkty již od roku 1970 a v současné době má téměř 3 300 zaměstnanců, ve světě je to pak 150 000 zaměstnanců ve více než 100 zemích. Vznikla v roce 1988 sjednocením firem ASEA a BBC, odtud název ABB. Součást ABB je také softwarové oddělení, ve kterém pracuje asi 170 zaměstnanců, z toho pak 30 v Ostravě. Softwarové oddělení se zabývá z velké části vývojem kontrolních systémů, které jsou například nasazovány v ropném průmyslu, vývojem tzv. Office Tools, což jsou například nástroje pro správu zaměstnanců a dalších informačních systémů, jež mají své využití v těžkém průmyslu.

2.2 Pracovní zařazení studenta

Počátkem srpna 2013 jsem se dostavil na přijímací pohovor, kde jsem prezentoval své dosavadní znalosti v oblasti programování, konkrétně na platformě .NET, v jazyce C#. Přijímacího pohovoru se účastnil jak manager sofwarového týmu, tak můj pozdější konzultant Radek Novák. Zde jsem byl obeznámen s projektem, na kterém bych měl pracovat. Jednalo se o programování instalátorů pro kontrolní aplikace, jež jsou nasazovány v ropném průmyslu.

Koncem srpna 2013 jsem tedy nastoupil na praxi, byl jsem seznámen s firemním prostředím a rovněž mi byli představeni další lidé z pracoviště, kde jsem praxi vykonával.

2.3 Harmonogram praxe

Časové rozpětí	**Popis**
19.9.2013 - 4.10.2013	Práce na instalátorech
10.10.2013 - 24.10.2013	Počátek vývoje Testing Tool, práce na uživatelském rozhraní
25.10.2013 - 31.10.2013	Studium aplikačních domén, jejich využití
1.11.2013 - 14.2.2014	Implementace importů pluginů do prostředí Testing Tool
20.2.2014 - 13.3.2014	Implementace projektů v Testing Tool
14.3.2013 - 21.3.2014	Refactoring Testing Tool

Tabulka 1: Časový harmonogram praxe

2.4 Hodiny strávené na jednotlivých úkolech

Úkol	**počet hodin**
Struktura instalátoru	32
Grafické rozhraní instalátoru	8
Celkem	**40**

Tabulka 2: Hodiny strávené na Instalátorech

Úkol	**počet hodin**
Objasnění projektu, návrh architektury	8
Grafické rozhraní Testing Tool	32
Studování aplikačních domén a jejích využití	20,5
Import pluginů - Plugin jako celek v aplikační doméně	64
Import pluginů - Metoda pomocí XML	8
Import pluginů - Metoda pomocí Pipeline	112,5
Import pluginů - Rozdělení logiky a uživatelského rozhraní pluginu	60
Projekty Testing Tool	55
Celkem	**360**

Tabulka 3: Hodiny strávené na projektu Testing Tool

3 Vývoj instalátorů pro aplikace

Má práce spočívala v programování instalátorů pro aplikace, jež jsou nasazovány v ropném průmyslu. Jedná se o rozsáhlé a komplexní aplikace, které sbírají a následně zpracovávají data z kontrolních systémů. Aplikace, pro který jsem měl instalátory programovat, je v podstatě balíček produktů, které slouží k monitorování alarm systémů a mým zadáním bylo udělat jeden velký instalátor, pro všechny tyto produkty. Aby tuto aplikaci bylo možné spustit na cílových počítačích a uživatel měl s nasazením aplikace co nejmenší starosti, musí být vytvořen jistý instalátor. Tento instalátor má zajistit podmínky pro správné fungování aplikace jako je například požadovaná verze .NET Framework, verze operačního systému atd. Na tuto problematiku jsem použil nástroj Windows Installer XML Toolset (WiX).

Na tomto projektu jsem pracoval pouze 7 dní. Jako důvod mi byl sdělen fakt, že instalátory již není nutné vyvíjet a bylo mi přichystáno téma nové, které zmíním níže v této práci.

3.1 Nástroj WiX toolset

Nástroj WiX Toolset je open source již od roku 2004. Jde vlastně o sadu nástrojů, které jsou schopny vytvářet instalační balíčky s příponou .msi a to pomocí XML jazyka. Pro tuto sadu existuje možnost doinstalovat ji do Visual Studia, což umožňuje využívat jeho vlastností jako je například IntelliSense a dále vstupní kód kompilovat s pomocí Visual Studio IDE.

3.2 Práce na instalátorech

Rád bych zdůraznil, že tato problematika byla pro mne zcela nová, tudíž jsem zpočátku získával opravdu ty nejzákladnější znalosti, které jsem poté v praxi implementoval s pomocí různých tutoriálů na serverech, které byly na tuto oblast vývoje zaměřeny nebo jsem požádal o pomoc svého konzultanta. Rovněž bych chtěl upozornit na fakt, že jsem nad touto problematikou strávil pouhých 7 dní a poté pracoval na novém projektu, na němž jsem strávil o mnoho více času, tudíž bude tato kapitola poněkud zobecněna.

K objasnění problematiky instalátorů mi pomohl web: [4]. V tomto tutoriálu jsem se naučil rozeznávat, z čeho se projekt skládá a jaké mají jednotlivé soubory v projektu roli. Zde jsem se rovněž dozvěděl, k čemu slouží tzv. MSI balíčky a z čeho se skládají.

3.2.1 Struktura instalátoru

Samotná instalace ve WiX se skládá z:

- Uživatelského prostředí instalace
- Lokalizace instalace
- Provádění customizací
- Kontrola spuštění, spojování modulů, vlastní operace
- Bootstrapper

V následujících krocích se pokusím alespoň v základu popsat, jak vypadá taková syntaxe kódu ve WiX.

Výpis kódu 1 nastiňuje, jak by měla vypadat definice instalace. Elementy <**?define?**> se používají na deklaraci proměnných, na které se lze v celém projektu odkazovat. Jako nejdůležitější část ukázkového kódu bych zmínil elementy <**Directory**>, které definují adresářovou strukturu, kam bude produkt instalován a element <**Component**>, který udává už přímo daný produkt, který bude instalován.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <? define ProductName = "Testingapplication" ?>
  <? define ProductVersion = "1.00.0000" ?>
  <? define ManufacturerName = "IMP" ?>
  <? define ManufacturerWebUrl ="http://www.imp.cz" ?>
  <? define DefaultInstallDir = "TestApplication" ?>
<Product Id="b557beb1-8852-42c2-a289-8e7336f3aa0b" Name="!(loc.ProductName
    )" Language="!(loc.Language)" Version="1.0.0.0" Manufacturer="
    MyFirstProject" UpgradeCode="9ba0e2f5-e02d-4fbb-a32d-c935fb338ab0"
    Codepage="0"> <Package InstallerVersion="200" Compressed="yes" />
<!--Check for OS Version-->
<Condition Message="!(loc.WindowsVersionRequired)">
 <![CDATA[VersionNT >= 501]]>
</Condition>
<!--Check for installed .NET Framework Version-->
<PropertyRef Id="NETFRAMEWORK40FULL" />
<!-- NETFRAMEWORK40FULL is set to 1 if the .NET Framework 4.0 full is
    installed -->
<Condition Message="!(loc.dotNetRequired)">
<![CDATA[Installed OR NETFRAMEWORK40FULL]]>
</Condition>
<Media Id="1" Cabinet="TestApplication.cab" EmbedCab="yes" />
<Binary Id="CustomInstaller" SourceFile="binary\CustomInstaller.exe" />
<Directory Id="TARGETDIR" Name="SourceDir">
     <Directory Id="ProgramFilesFolder">
       <Directory Id="APPLICATIONROOTDIRECTORY" Name="TestApplication">
         <Component Id ="TestApplication.exe" Guid="fe862ead-e74b-4098-af40
             -1bb4576e8b80">
           <File Id ="TestApplication.exe" Source="Release\TestApplication.
               exe" />
         </Component>
       </Directory>
     </Directory>
</Directory>
</Product>
</Wix>
```

Výpis 1: Definice instalátoru

Výše zmíněný kód ukazuje jakousi základní definici, co má takový instalátor umět - existuje určitý produkt (v tomto případě TestApplication.exe), který chceme nainstalovat.

Přitom jsou dány jisté podmínky pro cílový počítač tzn. verze .NETu, verze operačního systému atp.

3.2.2 Grafické rozhraní instalátoru

Instalace rovněž zahrnuje grafické rozhraní tak, aby bylo uživatelsky přijatelné. Samotný .NET zahrnuje knihovnu *WixUIExtension.dll*, ve které se nachází dialogová okna, jež jsou při instalaci zobrazována uživateli. Příklad, jak vypadá taková reference a následná konfigurace těchto dialogových oken je uvedeno ve výpisech kódu 2 a 3.

```
<DialogRef Id="BrowseDlg" />
<DialogRef Id="WelcomeDlg" />
```

Výpis 2: Reference na dialogová okna

```
<Publish Dialog="BrowseDlg"
Control="OK"
Event="DoAction"
Value="WixUIValidatePath"
Order="3">1</Publish>

<Publish Dialog="WelcomeDlg"
Control="Next"
Event="NewDialog"
Value="LicenseAgreementDlgModified">NOT Installed</Publish>
```

Výpis 3: Konfigurace dialogových oken

Podrobnější dokumentaci ke konfiguraci GUI instalátorů lze nalézt na serveru [1],

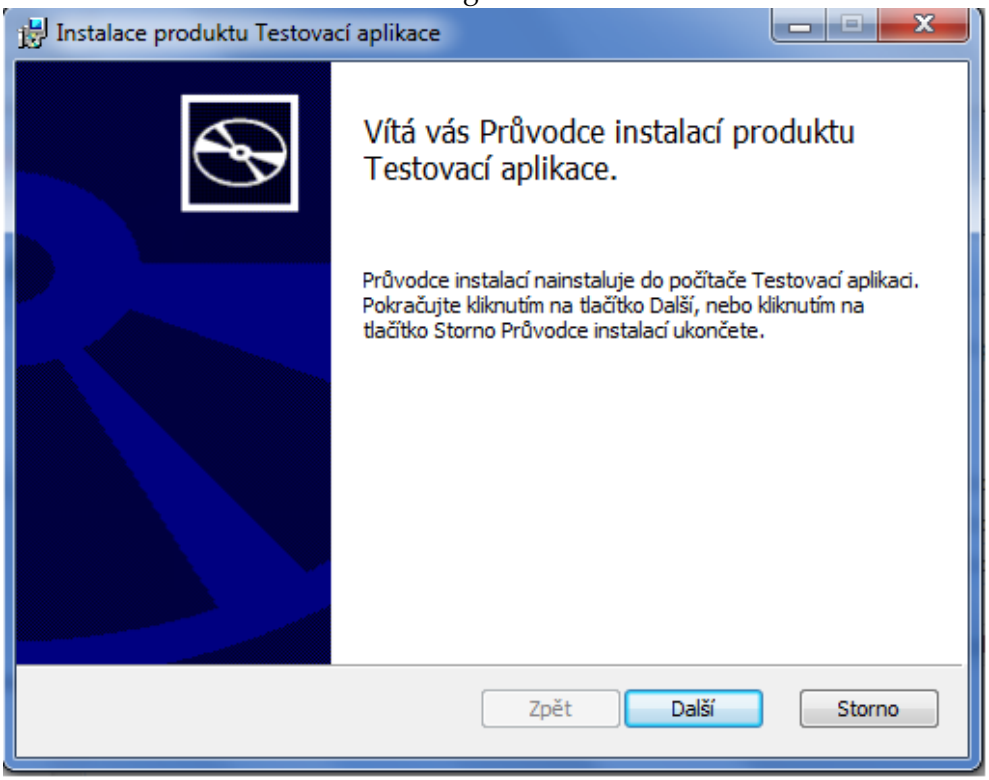
V projektu bylo třeba nastavit další podmínky jako je licenční ujednání instalace, dále jsem nakonfiguroval své vlastní dialogové okno "Licence Agreement", pro které jsem si vytvořil v projektu samostatný soubor *.wxs*, ve kterém jsem konfiguroval parametry jako je například změna polohy tlačítka pro potvrzení licenční smlouvy či změnu ikony obrazovky. Příklad, jak takový soubor může vypadat je prezentováno ve výpisu kódu 4. Elementy typu <**Control**> jsou vizuální prvky dialogového okna.

```
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
 <Fragment>
   <UI>
    <Dialog Id="LicenseAgreementDlgModified" Width="370" Height="270"
        Title="!(loc.LicenseAgreementDlg_Title)">

      <Control Id="BannerBitmap" Type="Bitmap" X="0" Y="0" Width="370"
          Height="44" TabSkip="no" Text="!(loc.
          LicenseAgreementDlgBannerBitmap)" />
      <Control Id="BannerLine" Type="Line" X="0" Y="44" Width="370" Height
          ="0" />
      <Control Id="BottomLine" Type="Line" X="0" Y="234" Width="370"
          Height="0" />
      <Control Id="Description" Type="Text" X="25" Y="23" Width="340"
          Height="15" Transparent="yes" NoPrefix="yes" Text="!(loc.
          LicenseAgreementDlgDescription)" />
      <Control Id="Title" Type="Text" X="15" Y="6" Width="200" Height="15"
           Transparent="yes" NoPrefix="yes" Text="!(loc.
          LicenseAgreementDlgTitle)" />
      <Control Id="LicenseAcceptedCheckBox" Type="CheckBox" X="20" Y="207"
           Width="271" Height="18" CheckBoxValue="1" Property="
          LicenseAccepted" Text="!(loc.
          LicenseAgreementDlgLicenseAcceptedCheckBox)" />
      <Control Id="Print" Type="PushButton" X="294" Y="207" Width="56"
          Height="17" Text="!(loc.WixUIPrint)">

    </Dialog>
   </UI>
</Fragment>
</Wix>
```

Výpis 4: Customizace dialogového okna

Po zkompilování projektu byl vytvořen soubor *MyFirstProject.msi*, který s doprovodnými dialogovými okny nainstaloval do adresáře *C:\Program Files\TestApplication* soubor s názvem *TestApplication.exe*. Pro ukázku, jak vypadá například dialogové okno "*WelcomeDlg*". je zobrazeno na obrázku 1.

Obrázek 1: Dialogové okno instalátoru



4 Testing Tool

Projekt *Testing Tool* je projekt, na kterém jsem pracoval od úplného začátku. Hlavní myšlenkou tohoto projektu je ulehčit práci s určitými pluginy, které mají také své grafické rozhraní a které například něco testují.

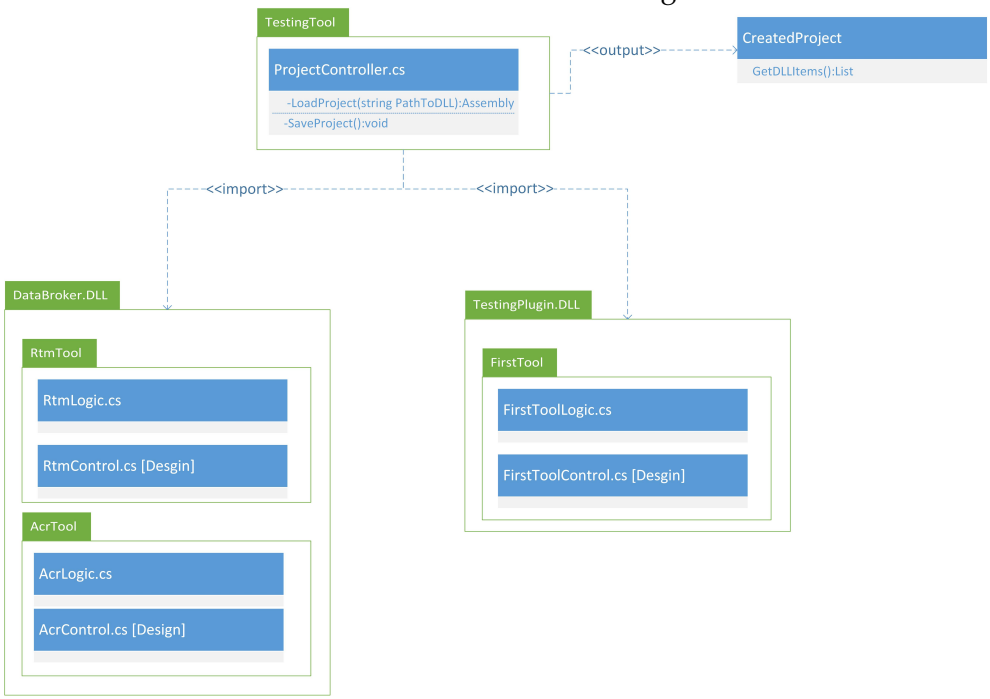
V průběhu vývoje Testing Tool jsem jako příklad používal testovací nástroj "DataBroker Emulator", jež má za úkol sbírat data z kontrolního systému a dále testovat jejich korektnost. Cílem tedy bylo, aby byl tento testovací nástroj nahrán do prostředí Testing Tool (jako .dll soubor) a poté v něm spuštěn. Testing Tool umožňuje tedy pluginy načíst a spustit ve svém prostředí. Z jednotlivých .dll souborů lze sestavit projekt, který je popsán v jazyce .xml a poté jej znovu spustit.

Projekt Testing Tool je vyvíjen pod technologií .NET v jazyce C#. Konkrétně ve WinForms.

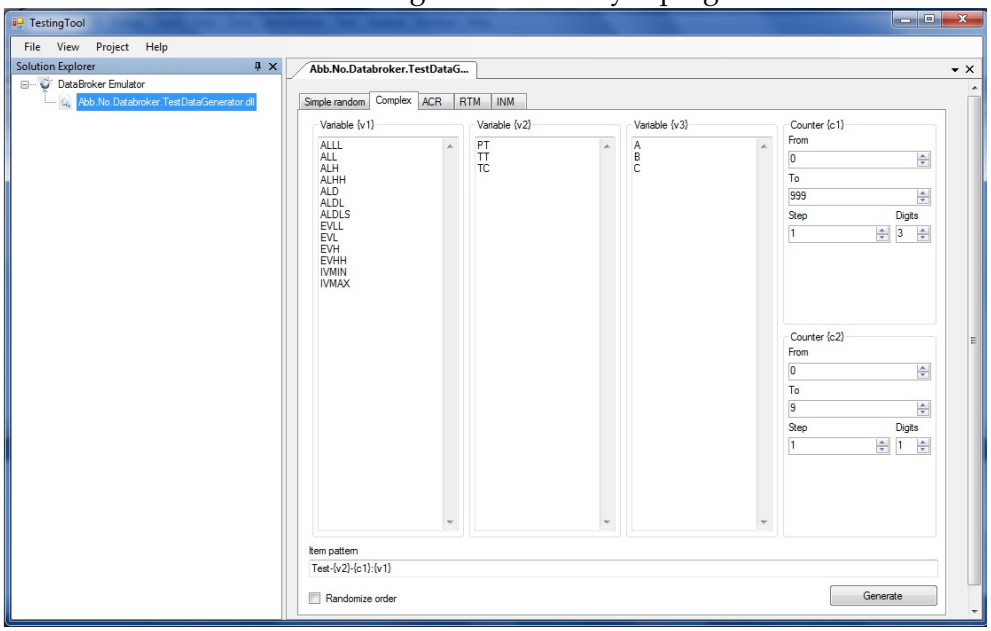
4.1 Architektura

Diagram na obrázku 2 vyjadřuje, jak Testing Tool pracuje. Základem je schopnost načíst .dll soubory, které musí mít určitou vnitřní strukturu, o níž se blíže zmíním později. Jak už bylo řečeno, každý takový .dll soubor má rovněž své grafické rozhraní, které musí být dostupné v Testing Toolu v plném rozsahu. Součástí Testing Toolu je také *ProjectController*, který se stará o založení, načítání a ukládání projektu prostřednictvím jazyka .xml.

Obrázek 2: Architektura TestingTool



Obrázek 3: Prostředí TestingToolu s načteným pluginem *DataBroker*



4.2 Grafické rozhraní Testing Tool

Uživatelské rozhraní Testing Tool jsem tedy vyvíjel ve WinForms. Všechny použité komponenty jsou tedy z jmenného prostoru *System.Windows.Forms*. Bylo ovšem nutné zajistit, aby určité komponenty, které nesly vlastnosti tzv. "oken", bylo možné dokovat podle potřeby uživatele. Jako příklad bych mohl uvést "Solution Explorer", ve kterém se nacházely právě spuštěné projekty. Má aktuální verze .NET Framework žádnou takovou komponentu neposkytovala, tudíž jsem ji musel vyhledat na internetu, konkrétně na severu [5], který mi poskytl knihovnu, jež implementovala vlastnosti dokovacího okna. Nakonec stačilo tuto knihovnu přidat do referencí projektu ve Visual Studiu a začít s vlastní implementací.

4.3 Import pluginů

Tato problematika mi zabrala nejvíce času, která mi rovněž dodala nejvíce zkušeností a tudíž bych se jí pokusil vysvětlit trochu podrobněji.

Docházelo ke komplikacím týkajících se serializace spojené s aplikačními doménami. Bylo totiž nutné, aby každý takový plugin (.dll soubor) měl svou vlastní aplikační doménu, přitom grafické rozhraní každého pluginu bylo reprezentováno v samotném Testing Toolu. Jednoduše řečeno bylo nutné dosáhnout toho, aby v aplikační doméně Testing Toolu bylo pouze grafické rozhraní pluginu, zatímco kód tohoto pluginu se nacházel ve své aplikační doméně. Co je to aplikační doména a proč je výše zmíněné rozdělení nutné, vysvětlím v kapitole 4.3.1 a v kapitole 4.3.2.

4.3.1 Co jsou aplikační domény

V dnešní době používají moderní operační systémy k izolaci mezi aplikacemi tzv. **procesy**. Nutnost izolovat aplikace je zavedena z toho důvodu, že všechny adresy proměnných, které daný proces využívá pro svou aplikaci, jsou relativní k tomuto procesu. Nejde tedy o absolutní adresu v paměti, z čehož plyne, že tyto proměnné nemohou být použity v žádném jiném cílovém procesu.

Aplikační domény mají za úkol rovněž izolovat aplikace, ale co je nejdůležitější je fakt, že **umožňují** komunikovat mezi jednotlivými aplikacemi. V jednom procesu může být spuštěno několik aplikačních domén, které by měly stejnou úroveň izolace, jako v oddělených procesech. Ovšem zde už bez jakéhokoliv přepínání mezi procesy.

Hlavními výhodami aplikačních domén jsou:

- Chyby v jedné aplikaci nemohou ovlivnit běh jiné aplikace
- Kód spuštěný v jedné aplikaci nemůže přímo přistupovat ke kódu nebo prostředkům jiné aplikace. Brání tomu tzv. **CLR (Common Language Runtime)**, který provádí kontrolu tzv. "Type safety code", což je v podstatě typově bezpečný kód a znamená to, že takový kód přistupuje pouze k paměti, ke které má přístup
- Chování kódu je určeno aplikací, ve které je práve spuštěn

4.3.2 Použití aplikačních domén

Z důvodu vzájemné nazávislosti pluginů a prostředí Testing Tool bylo nutné, aby byly rozděleny do dvou samostatných aplikačních domén grafické rozhraní a kód jednotlivých pluginů. Toto rozdělení již vyplývá z jedné z výhod, které aplikační doména nabízí - Chyby v jedné aplikaci nemohou ovlivnit běh jiné aplikace. Pokud je tedy určitý plugin nahrán do Testing Toolu, je nutné, aby veškeré výjimky jež byly vyvolány v daném pluginu, neohrozily popřípadě nezpůsobily "pád" celého Testing Toolu.

4.3.3 Postupy řešení importů pluginů

Pro práci s aplikačními doménami nabízí .NET třídu s názvem *System.AppDomain* a pro práci s assembly jmenný prostor *System.Reflection*, kde se nachází třída *Assembly*. Třídu *Assembly* jsem využíval pro import jednotlivých .dll souborů (pluginů) a třídu *System.AppDomain* pro instancování logiky pluginu v nové aplikační doméně. Ukázka využití těchto tříd se nachází ve výpisu kódu 5.

V následujících podkapitolách jsou uvedeny mé postupy řešení pro import jednotlivých pluginů do prostředí Testing Tool, jež jsem považoval za postupy se zdárným koncem. V podkapitolách 4.3.3.1 až 4.3.3.3 popisuji mé zvolené postupy řešením, které se nakonec ukázaly jako neúspěšné. Kapitola 4.3.4 pak obsahuje postup již se správným řešením.

4.3.3.1 Plugin jako celek v aplikační doméně

Jako prvotní řešení se mi zdálo vhodné načíst celý plugin (.DLL soubor) do samostatné aplikační domény. Tím bych si zajistil, že veškerý kód z tohoto .dll souboru se bude "odehrávat" mimo aplikační doménu Testing Toolu. Grafické rozhraní mělo být zkopírované do aplikační domény Testing Toolu tím způsobem, že veškeré komponenty typu *System.Windows.Controls* byly přeneseny do jeho uživatelského rozhraní. Ukázka kódu 5 prezentuje, jak bylo implementováno načítání assembly a výpis kódu 6 pak nastiňuje, jak bylo přeneseno uživatelské rozhraní pluginu.

```
Assembly pluginAssembly=Assembly.LoadFile(pathToDll);
AppDomain pluginDomain = AppDomain.CreateDomain("PluginDomain");
Form pluginForm=pluginDomain.CreateInstanceFromAndUnwrap(pluginAssmebly.Assembly.
    Location, typeof(Form).FullName) as Form;
```

Výpis 5: Načtení assembly

```
List<Control> pluginComponents=new List<Control>();
pluginComponents=pluginForm.Controls;
```

Výpis 6: Přenesení uživatelského rozhraní z pluginu

Zde ovšem nastal problém. Když jsem se pokoušel přenést prvky uživatelského rozhraní, tedy prvky typu *System.Windows.Controls*, z aplikační domény pluginu do aplikační domény Testing Toolu, byla během kompilace vyhozena výjimka typu *"System.Windows.Forms.Control is not marked as serializable"*. Tato výjimka v podstatě znamená, že prvky, jež dědí ze třídy *System.Windows.Controls* nejsou označeny jako serializovatelné, což má za následek, že je není možné "přenášet" skrze aplikační domény.

4.3.3.2 Metoda pomocí XML

Nabízelo se řešení popsat grafické rozhraní každého pluginu do dokumentu ve fomátu XML. Tento dokument by jednoduše obsahoval seznam prvků, jež jsou součástí grafického rozhraní. Testing Tool by pak nemusel nic serializovat a pouze by si popsané prvky instancoval ve své doméně. Tím by se ale pouze vyřešila grafická stránka pluginu. Musel bych najít způsob, jak odchytávat události typu "buttonClick()" atp., které by byly rovněž popsány v určitém dokumentu. To mi přišlo nakonec nevhodné, jelikož by se kód, který by na ony události reagoval, musel zpracovávat v jiné aplikační doméně.

4.3.3.3 Metoda Pipeline

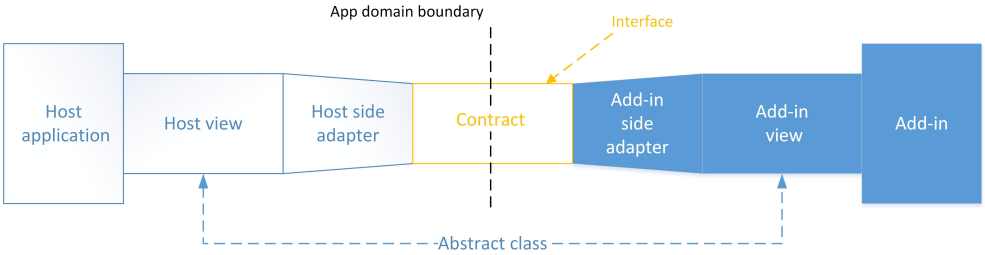
Od verze .NETu 3.5 je k dispozici jmenný prostor s názvem *System.AddIn*, který obsahuje veškeré třídy potřebné k implementaci "Pipeline" modelu.

Tento model je přímo určený pro práci s pluginy, resp. pro komunikaci mezi nimi a hostitelskou aplikací. Podstatné je ale to, že tato komunikace probíhá mezi aplikační doménou hostitelské aplikace a aplikační doménou jednotlivých pluginů. Základem tohoto modelu je jakýsi "řetěz" komponent, které se na této komunikaci podílí a které zároveň

určují pravidla, jakým způsobem bude komunikace probíhat. Cílem je, aby se veškeré metody volané v hostitelské aplikaci provedly pouze v aplikační doméně cílového pluginu a v hostitelské aplikaci pouze volaly. Na obrázku 4 je graficky popsán model.

Na jednom konci se nachází hostitelská aplikace a na opačném vždy daný plugin. Komponenty, jež se nachází mezi těmito konci se už podílejí na samotné komunikaci. V centru celé "Pipeline" se nachází komponenta *Contract*, což je v podstatě interface, ve kterém jsou nadefinovány metody, které musí být implementovány v pluginu a které jsou rovněž definovány s komponentách označených *Host side adapter* a *Add-in side adapter*, což jsou abstraktní třídy. Liší se však tím, že *Host side adapter* je označena atributem *System.AddIn.Pipeline.HostAdapterAttribute* a třída *Add-in side adapter* jako *System.AddIn.Pipeline.AddInAdapterAttribute*. Důležité je, že komponenta *Contract* se nachází na hranici aplikačních domén pluginů a hostitelské aplikace, čímž tvoří jakýsi přechod mezi oběma konci "Pipeline" .

Obrázek 4: Model Pipeline



Tímto jsem se pokusil nastínit, na jakém principu model "Pipeline" pracuje. Začal jsem tedy z implementací tohoto modelu přímo v Testing Toolu. Samotný Testing Tool jsem rozšířil o výše zmíněné komponenty. V interfacu *Contract* jsem definoval svou metodu, která tedy určovala, co musí daný plugin implementovat. Jednalo se o metodu, která vracela vizuální prvky daného pluginu, tedy prvky typu *System.Windows.Controls* viz. výpis kódu 7 a následná implementace v pluginu vypadala, jak naznačuje výpis kódu 8.

```
[AddInContract]
    public interface INumberProcessorContract : IContract
    {
        #region Methods
        List<System.Windows.Controls> GetAllPluginVisualComponents();
        void  Initialize  (IHostObjectContract hostObj);
        #endregion
    }
```

Výpis 7: Contract interface

```
public List<System.Windows.Controls> GetAllPluginVisualComponents()
{
 List<System.Windows.Controls> visualComponents=new List<System.Windows.Controls>();
            foreach (Control item in this.Controls)
            {
                visualComponents.Add(item);
            }

return visualComponents;
}
```

Výpis 8: Metoda pro získání vizuálních prvků

Mým cílem bylo tedy to, že bych z Testing Toolu zavolal metodu "GetAllPluginVisualComponents()", která by mi z pluginu, který se nachází ve své aplikační doméně, vrátila všechny vizuální prvky, které jsem si chtěl zobrazit v samotném Testing Toolu. Ovšem samotný kód, který by reagoval na události jednotlivých vizuálních prvků, by se spouštěl v aplikační doméně pluginu. Zde však nastal stejný problém, týkající se tzv. serializace prvků typu *System.Windows.Controls*. Když jsem zavolal v aplikační doméně Testing Toolu metodu "GetAllPluginVisualComponents()", tak se provedla v doméně pluginu, avšak problém nastal při návratové hodnotě, což je v tomto případě List prvků *System.Windows.Controls*.

Po delším zkoumání tohoto problému jsem se dozvěděl, že ony prvky typu *System.Windows.Controls* by serializovat šly, ovšem samotným problémem je opačná operace, tedy deserializace. Tyto prvky totiž obsahují mnoho "runtime" stavů a instancovat naprosto stejný objekt tohoto typu by znamenalo vysoké riziko výskytu chyby.

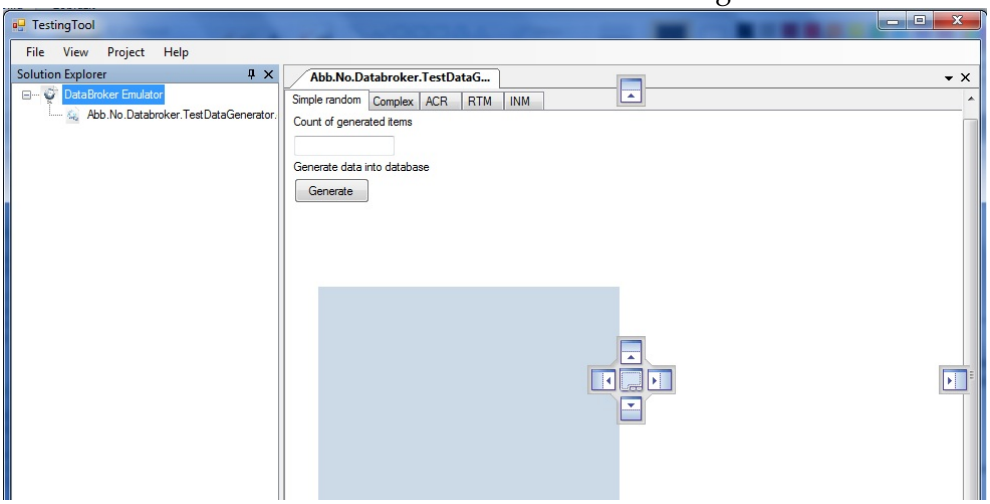
4.3.4 Rozdělení logiky a uživatelského rozhraní pluginu

Jako zdárné řešení se ukázala metoda rozdělením logiky a samotných vizuálních prvků už přímo v pluginu. Při instancování pluginu v Testing Toolu jsem rovněž instancoval logiku tohoto pluginu, avšak v jeho aplikační doméně. Vizuální prvky, jež byly součástí aplikační domény Testing Toolu pak už pouze volaly metody, které se ovšem spouštěly v doméně logiky pluginu.

4.4 Projekty Testing Tool

Do prostředí Testing Tool jsem rovněž implementoval možnost práci s projekty, jejíž součástí je "Solution Explorer", ve kterém jsou zobrazeny právě spuštěné projekty a k nim příslušné pluginy. Tento "Solution Explorer" má rovněž funkci dokovacího okna, jak je možné vidět na obrázku 5.

Obrázek 5: Možnost dokování oken Testing Toolu



Samotné projekty se skládají z jednotlivých pluginů, přičemž jeden projekt může obsahovat více než jden plugin. Existuje také možnost do již založeného projektu přidávat další pluginy. Všechny projekty jsou popsané v .XML souboru podobně jako tomu je například v projektech ve Visual Studiu, kdy jsou jednotlivé soubory projektu obsaženy v souboru *.csProj* (v případě projektu C#). V ukázkovém kódu 9 je zobrazen obsah .XML souboru, který obsahuje projekt s názvem "DataBroker Emulator" a dále plugin "Abb.No.Databroker.TestDataGenerator.dll", který je součástí projektu. Při načítání pak používám jazyk LINQ, ukázka kódu 10.

```
<?xml version="1.0" encoding="utf-8"?>
<project name="DataBroker Emulator">
  <items>
    < dllfile  path="..\WindowsFormsApplication1\Emulator_project\Abb.No.Databroker.
        TestDataGenerator.dll" />
  </items>
</project>
```

Výpis 9: Projekt v .XML souboru

```
public static List<string> ReadFromXMLAndGetPaths(XDocument xdoc) {

    List<string> listOfDLLPath = new List<string>();

    var pathsToDllFiles = from listOfPlugins in xdoc.Descendants("items")
                          select new
                          {
                              Children=listOfPlugins.Element(" dllfile ").Attribute("path"),
                          };

    foreach (var PathToDllFile in pathsToDllFiles)
    {
```

```
                string  DLLPath = PathToDllFile.Children.Value;
                listOfDLLPath.Add(DLLPath);
            }

            return listOfDLLPath;
        }
```

Výpis 10: LINQ dotaz na projekty v .XML souboru

4.4.1 Založení projektu

Součástí mnou implementované práce s projekty je také možnost založení nového projektu, což v podstatě obnáší to, že se musí projekt pojmenovat, vybrat jednotlivé pluginy, které mají být součástí nového projektu a nakonec vybrat příslušnou cestu, kde má být celý projekt uložen.

5 Uplatněné, získané a chybějící dovednosti a znalosti

Je třeba zmínit, že je určitý rozdíl mezi praxí a výukou na vysoké škole, alespoň co se týče způsobu získávání znalostí. Na této praxi jsem získal mnoho užitečných zkušeností a to nejen v oblasti programování jako takového, ale také v samostatnosti.

Jako uplatněné znalosti bych uvedl základy jazyka C#, znalosti OOP (Objektově orientovaného programování), jež byly nutnou podmínkou pro nástup do praxe a v neposlední řadě znalosti jazyka XML. V průběhu letního semestru 3. ročníku jsem rovněž uplatňoval znalosti získané během studia. Naučil jsem se používat tzv. "anonymní metody" či "lambda výrazy", které poměrně zpřehlednily zdrojový kód.

Za opravdu přínosné považuji znalosti získané na platformě .NET a rovněž programovacím jazyce C#. Opravdu do hloubky jsem si prostudoval, na jakém principu pracují aplikační domény a rovněž jsem si procvičil znalosti z oblasti reflexe. Jako přínosné a doufám že i do budoucna použitelné pokládám nahlédnutí do oblasti instalátorů.

Co se týče chybějících znalostí, tak bych mohl zmínit jazyk LINQ se kterým jsem se setkal až na této praxi. Ovšem celkově bych řekl, že každý den, který jsem na praxi docházel, jsem se musel něčemu novému přiučit.

6 Závěr

V průběhu praxe jsem zdokonalil své schopnosti v oblasti programování na platformě .NET. Setkal jsem se s problematikou aplikačních domén, které hrály velice důležitou roli v mém projektu Testing Tool. Abych byl schopný vyřešit úkoly týkající se importů pluginů a jejich následnému spuštění v prostředí Testing Tool, musel jsem si důkladně nastudovat, jak aplikační domény fungují a hlavně, jak s nimi zacházet. Jako přínosné považuji také práci s jazykem XML, kterým jsem byl schopen vytvářet strukturu projektů v Testing Tool, které hrály v tomto prostředí Testing Tool důležitou roli. Rovněž mi tato praxe umožnila získat alespoň základní představu o tom, co obnáší programování instalátorů a zároveň si jednoduchý instalátor naprogramovat.

Jsem velice rád, že mi byla poskytnuta možnost vykonávat bakalářskou praxi ve firmě ABB s.r.o. Mým hlavním cílem, než jsem nastoupil do praxe, bylo získat zkušenosti, jak to chodí ve firmě z oboru IT, což bylo splněno v plné míře. Naopak se mi naskytla možnost pokračovat ve spolupráci s touto firmou a podílet se na interním projektu, čehož se velice vážím. Myslím si, že má účast na této praxi bude mít pozitivní dopad co se mé budoucnosti týče, jelikož zkušenosti z praxe jsou nesmírně cenné a doufám, že se budu i nadále v tomto oboru zdokonalovat.

Jiří Draška

7 Reference

[1] WiX, *Windows Installer XML toolset*, [ONLINE].
URL: http://wixtoolset.org//

[2] Microsoft Inc., *Microsoft Development Network*, [ONLINE].
URL: http://msdn.microsoft.com/default.aspx#fbid=76h5OrpaIiG

[3] ABB Group, *The ABB Group a Automation and Power Technologies*, [ONLINE].
URL: http://new.abb.com/about/abb-in-brief

[4] DotNetPortal, *dotNET Portal*, [ONLINE].
URL: http://www.dotnetportal.cz/blogy/15/Null-Reference-Exception/5219/WiX-Jak-na-instalace-cast-1-Uvod

[5] CodeProject, *Visual Studio IDE like Dock Container*, [ONLINE].
URL: http://www.codeproject.com/Articles/25976/Visual-Studio-IDE-like-Dock-Container/